δ-Complete Analysis for Bounded Reachability of Hybrid Systems

Sicun Gao Soonho Kong Wei Chen
Edmund M. Clarke

July 16, 2014
CMU-CS-14-111

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

We present the framework of δ-complete analysis for bounded reachability problems of general hybrid systems. We perform bounded reachability checking through solving δ-decision problems over the reals. The techniques take into account of robustness properties of the systems under numerical perturbations. We prove that the verification problems become much more mathematically tractable in this new framework. Our implementation of the techniques, an open-source tool dReach, scales well on several highly nonlinear hybrid system models that arise in biomedical and robotics applications.

This research was sponsored by the National Science Foundation grants no. CNS1330014, no. CNS0926181 and no. CNS0931985, the GSRC under contract no. 1041377, the Semiconductor Research Corporation under contract no. 2005TJ1366, and the Office of Naval Research under award no. N000141010188.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
16 JUL 2014		**00-00-2014 to 00-00-2014**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
8 -Complete Analysis for Bounded Reachability of Hybrid Systems	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Carnegie Mellon University,School of Computer Science,Pittsburgh,PA,15213**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

We present the framework of -complete analysis for bounded reachability problems of general hybrid systems. We perform bounded reachability checking through solving -decision problems over the reals. The techniques take into account of robustness properties of the systems under numerical perturbations. We prove that the verification problems become much more mathematically tractable in this new framework. Our implementation of the techniques, an open-source tool dReach, scales well on several highly nonlinear hybrid system models that arise in biomedical and robotics applications.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**Same as Report (SAR)**	**19**	

Keywords: Hybrid Systems, Reachability, Nonlinear Systems

1 Introduction

Formal verification is difficult for hybrid systems with nonlinear dynamics and complex discrete controls [2,19]. A major difficulty of applying advanced verification techniques in this domain comes from the need of solving logic formulas over the real numbers with nonlinear functions, which is notoriously hard. Recently, we have defined the δ-*decision problem* that is much easier to solve [13,12]. Given an arbitrary positive rational number δ, the δ-decision problem asks if a logic formula is false or δ-*true* (or, dually, true or δ-*false*). The latter answer can be given, if the formula *would be true* under δ-bounded numerical changes on its syntactic form [13]. The δ-decision problem is decidable for bounded first-order sentences over the real numbers with arbitrary Type 2 computable functions. Type 2 computable functions [26] are essentially real functions that can be approximated numerically. They cover almost all functions that can occur in realistic hybrid systems, such as polynomials, trigonometric functions, and solutions of Lipschitz-continuous ODEs. The goal of this paper is to develop a new framework for solving bounded reachability problems for hybrid systems based on solving δ-decisions. We prove that this framework makes bounded reachability of hybrid systems a much more mathematically tractable problem and show that our practical implementation can handle highly nonlinear hybrid systems.

The framework of δ-*complete analysis* consists of techniques that perform verification and allow bounded errors on the safe side. For bounded reachability problems, δ-complete analysis aims to find one of the following answers:

- safe (bounded): The system does not violate the safety property within a bounded period of time and a bounded number of discrete mode changes.
- δ-unsafe: The system would violate the safety property under some δ-bounded numerical perturbations.

Thus, when the answer is safe, no error is involved. On the other hand, a system that is δ-unsafe would violate the safety property under bounded numerical perturbations. Realistic hybrid systems interact with the physical world and it is impossible to avoid slight perturbations. Thus, δ-unsafe systems should indeed be regarded as unsafe, under reasonable choices of δ. Note that such robustness problems can not be discovered by solving the precise decision problem, and the use of δ-decisions strengthens the verification results.

δ-Complete reachability analysis reduces verification problems to δ-decision problems of formulas over the reals. It follows from δ-decidability of these formulas [13] that δ-complete reachability analysis of a wide range of nonlinear hybrid systems is decidable. Such results stand in sharp contrast to the standard high undecidability of bounded reachability for simple hybrid systems.

We emphasize that the new framework is immediately practical. We implemented the techniques in our open-source tool dReach based on our nonlinear SMT solver dReal [14]. In our previous work, we have shown the underlying solver scales on nonlinear systems [15]. The tool successfully verified safety properties of various nonlinear models that are beyond the scope of existing tools.

The paper is organized as follows. After a short review of δ-decidability, we show how to represent hybrid systems with $\mathcal{L}_{\mathbb{R}_\mathcal{F}}$-formulas and how to interpret trajectories through semantics of the formulas in Section 2. Then we focus on bounded reachability and show the encoding in

$\mathcal{L}_{\mathbb{R}_\mathcal{F}}$ in Section 3. In Section 4, we show experimental results of our open-source implementation on highly nonlinear hybrid systems, and discuss the comparison with reachable set computation techniques in Section 5 and conclude in Section 6.

Related Work. Our framework can be seen as a converging point for several lines of existing work. First of all, the use of logic formulas to express model checking of hybrid systems dates back to [3,5], where formulas with linear arithmetic over the reals are used. The lack of an appropriate logic for encoding nonlinear systems beyond real arithmetic has been a major bottleneck in this direction. Second, the realization that robustness assumptions help reduce verification complexity as been realized frequently. Franzle's work [10] was among the first to recognize that verification problems are more tractable when robustness is assumed for polynomial hybrid systems. The direction was continued with more positive results such as [25]. These works present theoretical results that do not directly translate to practical solving techniques, and the results are sensitive to the definitions. For instance, it is also shown in [20] that a slightly different notion of robustness and noise does not improve the theoretical properties. We focus on formulating a framework that directly corresponds to practical solving techniques, and the positive theoretical results follow naturally at the same time. There has also been much recent work on using constraint solving techniques for solving hybrid systems [11,21,18,7], as well as solving frameworks that exploit robustness properties of the systems [24,22]. These methods can all handle nonlinear dynamics to certain degrees (mostly polynomial systems, with the exception of [7] which we will mention again in the experiments). We aim to extend these works to a most broad class of nonlinear hybrid systems, and provide precise correctness guarantees. We also provide an open-source implementation that scales well on highly nonlinear systems that arise in practical applications.

2 $\mathcal{L}_{\mathbb{R}_\mathcal{F}}$-Representations of Hybrid Automata

2.1 $\mathcal{L}_{\mathbb{R}_\mathcal{F}}$-Formulas and δ-Decidability

We will use a logical language over the real numbers that allows arbitrary *computable real functions* [26]. We write $\mathcal{L}_{\mathbb{R}_\mathcal{F}}$ to represent this language. Intuitively, a real function is computable if it can be numerically simulated up to an arbitrary precision. For the purpose of this paper, it suffices to know that almost all the functions that are needed in describing hybrid systems are Type 2 computable, such as polynomials, exponentiation, logarithm, trigonometric functions, and solution functions of Lipschitz-continuous ordinary differential equations.

More formally, $\mathcal{L}_{\mathbb{R}_\mathcal{F}} = \langle \mathcal{F}, > \rangle$ represents the first-order signature over the reals with the set $\mathcal{F}$ of computable real functions, which contains all the functions mentioned above. Note that constants are included as 0-ary functions. $\mathcal{L}_{\mathbb{R}_\mathcal{F}}$-formulas are evaluated in the standard way over the structure $\mathbb{R}_\mathcal{F} = \langle \mathbb{R}, \mathcal{F}^\mathbb{R}, >^\mathbb{R} \rangle$. It is not hard to see that we can put any $\mathcal{L}_{\mathbb{R}_\mathcal{F}}$-formula in a normal form, such that its atomic formulas are of the form $t(x_1, ..., x_n) > 0$ or $t(x_1, ..., x_n) \geq 0$, with $t(x_1, ..., x_n)$ composed of functions in $\mathcal{F}$. To avoid extra preprocessing of formulas, we can explicitly define $\mathcal{L}_\mathcal{F}$-formulas as follows.

Definition 1 (**$\mathcal{L}_{\mathbb{R}_\mathcal{F}}$-Formulas).** *Let $\mathcal{F}$ be a collection of computable real functions. We define:*

$$t := x \mid f(t(\boldsymbol{x})), \text{ where } f \in \mathcal{F} \text{ (constants are 0-ary functions)};$$
$$\varphi := t(\boldsymbol{x}) > 0 \mid t(\boldsymbol{x}) \geq 0 \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \exists x_i \varphi \mid \forall x_i \varphi.$$

In this setting $\neg\varphi$ is regarded as an inductively defined operation which replaces atomic formulas $t > 0$ with $-t \geq 0$, atomic formulas $t \geq 0$ with $-t > 0$, switches $\wedge$ and $\vee$, and switches $\forall$ and $\exists$.

Definition 2 (**Bounded $\mathcal{L}_{\mathbb{R}_\mathcal{F}}$-Sentences).** *We define the bounded quantifiers $\exists^{[u,v]}$ and $\forall^{[u,v]}$ as $\exists^{[u,v]}x.\varphi =_{df} \exists x.(u \leq x \wedge x \leq v \wedge \varphi)$ and $\forall^{[u,v]}x.\varphi =_{df} \forall x.((u \leq x \wedge x \leq v) \rightarrow \varphi)$ where u and v denote $\mathcal{L}_{\mathbb{R}_\mathcal{F}}$ terms, whose variables only contain free variables in φ excluding x. A* bounded $\mathcal{L}_{\mathbb{R}_\mathcal{F}}$-sentence *is*

$$Q_1^{[u_1,v_1]}x_1 \cdots Q_n^{[u_n,v_n]}x_n \ \psi(x_1, ..., x_n),$$

where $Q_i^{[u_i,v_i]}$ are bounded quantifiers, and $\psi(x_1, ..., x_n)$ is quantifier-free.

Definition 3 (**δ-Variants).** *Let $\delta \in \mathbb{Q}^+ \cup \{0\}$, and φ an $\mathcal{L}_{\mathbb{R}_\mathcal{F}}$-formula*

$$\varphi : \ Q_1^{I_1}x_1 \cdots Q_n^{I_n}x_n \ \psi[t_i(\boldsymbol{x}, \boldsymbol{y}) > 0; t_j(\boldsymbol{x}, \boldsymbol{y}) \geq 0],$$

where $i \in \{1, ...k\}$ and $j \in \{k+1, ..., m\}$. The δ-weakening *φ^δ of φ is defined as the result of replacing each atom $t_i > 0$ by $t_i > -\delta$ and $t_j \geq 0$ by $t_j \geq -\delta$:*

$$\varphi^\delta : \ Q_1^{I_1}x_1 \cdots Q_n^{I_n}x_n \ \psi[t_i(\boldsymbol{x}, \boldsymbol{y}) > -\delta; t_j(\boldsymbol{x}, \boldsymbol{y}) \geq -\delta].$$

It is clear that $\varphi \rightarrow \varphi^\delta$ (see [13]).

In [13,12], we have proved that the following δ-decision problem is decidable, which is the basis of our framework.

Theorem 1 (**δ-Decidability).** *Let $\delta \in \mathbb{Q}^+$ be arbitrary. There is an algorithm which, given any bounded $\mathcal{L}_{\mathbb{R}_\mathcal{F}}$-sentence φ, correctly returns one of the following two answers:*

- δ-True*: φ^δ is true.*
- False*: φ is false.*

When the two cases overlap, either answer is correct.

Theorem 2 (**Complexity).** *Let S be a class of $\mathcal{L}_{\mathbb{R}_\mathcal{F}}$-sentences, such that for any φ in S, the terms in φ are in Type 2 complexity class* C*. Then, for any $\delta \in \mathbb{Q}^+$, the δ-decision problem for bounded Σ_n-sentences in S is in $(\Sigma_\mathsf{n}^\mathsf{P})^\mathsf{C}$.*

2.2 $\mathcal{L}_{\mathbb{R}_\mathcal{F}}$-Representations and Hybrid Trajectories

We first show that $\mathcal{L}_{\mathbb{R}_\mathcal{F}}$-formulas can concisely represent hybrid automata.

Definition 4 ($\mathcal{L}_{\mathbb{R}_\mathcal{F}}$-Representations of Hybrid Automata). *A hybrid automaton in $\mathcal{L}_{\mathbb{R}_\mathcal{F}}$-representation is a tuple*

$$H = \langle X, Q, \{\mathsf{flow}_q(\boldsymbol{x}, \boldsymbol{y}, t) : q \in Q\}, \{\mathsf{inv}_q(\boldsymbol{x}) : q \in Q\}, \\ \{\mathsf{jump}_{q\to q'}(\boldsymbol{x}, \boldsymbol{y}) : q, q' \in Q\}, \{\mathsf{init}_q(\boldsymbol{x}) : q \in Q\}\rangle$$

where $X \subseteq \mathbb{R}^n$ for some $n \in \mathbb{N}$, $Q = \{q_1, ..., q_m\}$ is a finite set of modes, and the other components are finite sets of quantifier-free $\mathcal{L}_{\mathbb{R}_\mathcal{F}}$-formulas.

Notation 3 *For any hybrid system H, we write $X(H)$, $\mathsf{flow}(H)$, etc. to denote its corresponding components.*

Almost all hybrid systems studied in the existing literature can be defined by restricting the set of functions $\mathcal{F}$ in the signature. For instance,

Example 1 (Linear and Polynomial Hybrid Automata). Let $\mathcal{F}^{\text{lin}} = \{+\} \cup \mathbb{Q}$ and $\mathcal{F}^{\text{poly}} = \{\times\} \cup \mathcal{F}^{\text{lin}}$. Rational numbers are considered as 0-ary functions. In existing literature, H is a *linear hybrid automaton* if it has an $\mathcal{L}_{\mathbb{R}_{\mathcal{F}^{\text{lin}}}}$-representation, and a *polynomial hybrid automaton* if it has an $\mathcal{L}_{\mathbb{R}_{\mathcal{F}^{\text{poly}}}}$-representation.

Example 2 (Nonlinear Bouncing Ball). The bouncing ball is a standard hybrid system model. Its nonlinear version (with air drag) can be $\mathcal{L}_{\mathbb{R}_\mathcal{F}}$-represented in the following way:

- $X = \mathbb{R}^2$ and $Q = \{q_u, q_d\}$. We use q_u to represent bounce-back mode and q_d the falling mode.
- $\mathsf{flow} = \{\mathsf{flow}_{q_u}(x_0, v_0, x_t, v_t, t), \mathsf{flow}_{q_d}(x_0, v_0, x_t, v_t, t)\}$. We use x to denote the height of the ball and v its velocity. Instead of using time derivatives, we can directly write the flows as integrals over time, using $\mathcal{L}_{\mathbb{R}_\mathcal{F}}$-formulas:
  - $\mathsf{flow}_{q_u}(x_0, v_0, x_t, v_t, t)$ defines the dynamics in the bounce-back phase:

$$(x_t = x_0 + \int_0^t v(s)ds) \wedge (v_t = v_0 + \int_0^t g(1 - \beta v(s)^2)ds)$$

  - $\mathsf{flow}_{q_d}(x_0, v_0, x_t, v_t, t)$ defines the dynamics in the falling phase:

$$(x_t = x_0 + \int_0^t v(s)ds) \wedge (v_t = v_0 + \int_0^t g(1 + \beta v(s)^2)ds)$$

  where β is a constant. Again, note that the integration terms define Type 2 computable functions.
- $\mathsf{jump} = \{\mathsf{jump}_{q_u\to q_d}(x, v, x', v'), \mathsf{jump}_{q_d\to q_u}(x, v, x', v')\}$ where
  - $\mathsf{jump}_{q_u\to q_d}(x, v, x', v')$ is $(v = 0 \wedge x' = x \wedge v' = v)$.
  - $\mathsf{jump}_{q_d\to q_u}(x, v, x', v')$ is $(x = 0 \wedge v' = \alpha v \wedge x' = x)$, for some constant α.
- init_{q_d} is $(x = 10 \wedge v = 0)$ and init_{q_u} is $\bot$.
- inv_{q_d} is $(x >= 0 \wedge v >= 0)$ and inv_{q_u} is $(x >= 0 \wedge v <= 0)$.

Trajectories of hybrid systems combine continuous flows and discrete jumps. This motivates the use of a hybrid time domain, with which we can keep track of both the discrete changes and the duration of each continuous flow. A hybrid time domain is a sequence of closed intervals on the real line, and a hybrid trajectory is a mapping from the time domain to the Euclidean space. Formally, we use the following definition given by Davoren in [9]:

Definition 5 (Hybrid Time Domains and Hybrid Trajectories [9]). *A* hybrid time domain *is a subset of* $\mathbb{N} \times \mathbb{R}$ *of the form*

$$T_m = \{(i,t) : i < m \text{ and } t \in [t_i, t_i'] \text{ or } [t_i, +\infty)\},$$

where $m \in \mathbb{N} \cup \{+\infty\}$, $\{t_i\}_{i=0}^{m}$ *is an increasing sequence in* $\mathbb{R}^+$, $t_0 = 0$, *and* $t_i' = t_{i+1}$. *When* $X \subseteq \mathbb{R}^n$ *is an Euclidean space and* T_m *a hybrid time domain, a* hybrid trajectory *is a* continuous *mapping* $\xi : T_m \to X$. *We can write the time domain* T_m *of* ξ *as* $T(\xi)$.

We can now define trajectories of hybrid automata. To link hybrid trajectories with automata, we need a labeling function $\sigma_{\xi,H}(i)$ that maps each step i in the hybrid trajectory to an appropriate discrete mode in H, and make sure that the $\mathsf{flow}, \mathsf{jump}, \mathsf{inv}, \mathsf{init}$ conditions are satisfied.

Definition 6 (Trajectories of Hybrid Automata). *Let* H *be a hybrid automaton,* T_m *a hybrid domain, and* $\xi : T_m \to X$ *a hybrid trajectory. We say that* ξ *is* a trajectory of H of discrete depth m, *written as* $\xi \in [\![H]\!]$, *if there exists a* labeling function $\sigma_{\xi,H} : \mathbb{N} \to Q$ *such that:*

- *For some* $q \in Q$, $\sigma_{\xi,H}(0) = q$ *and* $\mathbb{R}_{\mathcal{F}} \models \mathsf{init}_q(\xi(0,0))$.
- *For any* $(i,t) \in T_m$, $\mathbb{R}_{\mathcal{F}} \models \mathsf{inv}_{\sigma_{\xi,H}(i)}(\xi(i,t))$.
- *For any* $(i,t) \in T_m$,
  - *When* $i = 0$, $\mathbb{R}_{\mathcal{F}} \models \mathsf{flow}_{q_0}(\xi(0,0), \xi(0,t), t)$.
  - *When* $i = k+1$, *where* $0 < k+1 < m$,

$$\mathbb{R}_{\mathcal{F}} \models \mathsf{flow}_{\sigma_\xi^H(k+1)}(\xi(k+1,t_{k+1}), \xi(k+1,t), (t - t_{k+1})), \text{and}$$
$$\mathbb{R}_{\mathcal{F}} \models \mathsf{jump}_{\sigma_{\xi,H}(k) \to \sigma_{\xi,H}(k+1)}(\xi(k,t_k'), \xi(k+1,t_{k+1})).$$

The definition is straightforward. In each mode, the system flows continuously following the dynamics defined by flow_q. Note that $(t - t_k)$ is the actual duration in the k-th mode. When a switch between two modes is performed, it is required that $\xi(k+1, t_{k+1})$ is updated from the exit value $\xi(k, t_k')$ in the previous mode, following the jump conditions.

Remark 1 (jump *vs* inv*).* The jump conditions specify when H *may* switch to another mode. The invariants (when violated) specify when H *must* switch to another mode. They will require different logical encodings.

Note that we gave no restriction on the formulas that can be used for describing hybrid automata in Definition 4. A minimal requirement is that the flow predicates should define continuous trajectories over time, namely:

Definition 7 (Well-Defined Flow Predicates). *Let* $\mathsf{flow}(\boldsymbol{x}, \boldsymbol{y}, t)$ *be a flow predicate for a hybrid automaton* H. *We say the flow predicate is* well-defined, *if for all tuples* $(\boldsymbol{a}, \boldsymbol{b}, \tau) \in X(H) \times X(H) \times \mathbb{R}^{\geq 0}$ *such that* $\mathbb{R} \models \mathsf{flow}(\boldsymbol{a}, \boldsymbol{b}, \tau)$, *there exists a continuous function* $\eta : [0, \tau] \to X$ *such that* $\eta(0) = \boldsymbol{a}$, $\eta(\tau) = \boldsymbol{b}$, *and for all* $t' \in [0, \tau]$, *we have* $\mathbb{R} \models \mathsf{flow}(\boldsymbol{a}, \eta(t), t)$. *We say* H *is* well-defined *if all its flow predicates are well-defined.*

This definition requires that we can always construct a trajectory from the end points and the initial points that satisfy a flow predicate. Flows that are defined using differential equations, differential inclusions, and explicit continuous mappings all satisfy this condition. Thus, from now on our discussion of hybrid automata assume their well-definedness.

2.3 δ-Perturbations

We can now define δ-perturbations on hybrid automata directly through perturbations on the logic formulas in their $\mathcal{L}_{\mathbb{R}_\mathcal{F}}$-representations. For any set S of $\mathcal{L}_{\mathbb{R}_\mathcal{F}}$-formulas, we write S^δ to denote the set containing the δ-perturbations of all elements of S.

Definition 8 (δ-Weakening of Hybrid Automata). *Let $\delta \in \mathbb{Q}^+ \cup \{0\}$ be arbitrary. Suppose*

$$H = \langle X, Q, \mathsf{flow}, \mathsf{jump}, \mathsf{inv}, \mathsf{init} \rangle$$

is an $\mathcal{L}_{\mathbb{R}_\mathcal{F}}$-representation of hybrid system H. The δ-weakening *of H is*

$$H^\delta = \langle X, Q, \mathsf{flow}^\delta, \mathsf{jump}^\delta, \mathsf{inv}^\delta, \mathsf{init}^\delta \rangle$$

which is obtained by weakening all formulas in the $\mathcal{L}_{\mathbb{R}_\mathcal{F}}$-representations of H.

Example 3. The δ-weakening of the bouncing ball automaton is obtained by weakening the formulas in its description. For instance, $\mathsf{flow}^\delta_{q_u}(x_0, v_0, x_t, v_t, t)$ is

$$|x_t - (x_0 + \int_0^t v(s)ds)| \leq \delta \wedge |v_t - (v_0 + \int_0^t g(1 - \beta v(s)^2)ds))| \leq \delta$$

and $\mathsf{jump}^\delta_{q_d \to q_u}(x, v, x', v')$ is

$$|x| \leq \delta \wedge |v' - \alpha v| \leq \delta \wedge |x' - x| \leq \delta.$$

Remark 2. It is important to note that the notion of δ-perturbations is a purely syntactic one (defined on the description of hybrid systems) instead of a semantic one (defined on the trajectories). The syntactic perturbations correspond to semantic over-approximation of H in the trajectory space.

Proposition 1. *For any H and $\delta \in \mathbb{Q}^+ \cup \{0\}$, $[\![H]\!] \subseteq [\![H^\delta]\!]$.*

Proof. Let $\xi \in [\![H]\!]$ be any trajectory of H. Following Definition 3, for any $\mathcal{L}_{\mathbb{R}_\mathcal{F}}$ sentence φ, we have $\varphi \to \varphi^\delta$. Since ξ satisfies the conditions in Definition 6, after replacing each formula by their δ-weakening, we have $\xi \in [\![H^\delta]\!]$.

2.4 Reachability

We can now formally state the reachability problem for hybrid automata using $\mathcal{L}_{\mathbb{R}_\mathcal{F}}$-representations and their interpretations.

Definition 9 (Reachability). *Let H be an n-dimensional hybrid automaton, and U a subset of its state space $Q \times X$. We say U* is reachable by *H, if there exists $\xi \in [\![H]\!]$,such that there exists $(i,t) \in T(\xi)$ satisfying $(\sigma_\xi^H(i), \xi(i,t)) \in U$.*

The bounded reachability problem for hybrid systems is defined by restricting the continuous time duration to a bounded interval, and the number of discrete transitions to a finite number.

Definition 10 (Bounded Reachability). *Let H be an n-dimensional hybrid automaton, whose continuous state space X is a bounded subset of $\mathbb{R}^n$. Let U be a subset of its state space. Set $k \in \mathbb{N}$ and $M \in \mathbb{R}^{\geq 0}$. The (k, M)-bounded reachability problem asks whether there exists $\xi \in [\![H]\!]$ such that there exists $(i,t) \in T(\xi)$ with $i \leq k$, $t = \sum_{i=0}^{k} t_i$ where $t_i \leq M$, and $(\sigma_\xi(i), \xi(i,t)) \in U$.*

Remark 3. By "step", we mean the number of discrete jumps. We say H can reach U in k steps, if there exists $\xi \in [\![H]\!]$ that contains k discrete jumps, which consists of $k+1$ pieces of continuous flows in the corresponding discrete modes.

In the seminal work of [4,3], it is already shown that the bounded reachability problem for simple classes of hybrid automata is undecidable. The goal of δ-complete analysis is to bypass much of this difficulty.

3 δ-Complete Analysis for Bounded Reachability

3.1 Encoding Bounded Reachability in $\mathcal{L}_{\mathbb{R}_\mathcal{F}}$

We now define the $\mathcal{L}_{\mathbb{R}_\mathcal{F}}$-encoding of bounded reachability. The encodings are standard bounded model checking, and have been studied in existing work but without the generality of a full $\mathcal{L}_{\mathbb{R}_\mathcal{F}}$-language. As a result, some issues have not been discovered. For example, the full encoding of non-deterministic flows with invariant conditions require second-order quantification, and the first-order encoding requires additional assumptions. We will give the full $\mathcal{L}_{\mathbb{R}_\mathcal{F}}$-encodings and discuss such details.

Notation 4 *Let H be a hybrid automaton. We use $\mathsf{unsafe} = \{\mathsf{unsafe}_q : q \in Q\}$ as the $\mathcal{L}_{\mathbb{R}_\mathcal{F}}$-representation of an unsafe region in the state space of H. We can write $[\![\mathsf{unsafe}]\!] = \bigcup_{q \in Q} [\![\mathsf{unsafe}_q]\!] \times \{q\}$.*

First, we need to define a set of auxiliary formulas that will be important for ensuring that a particular mode is picked at a certain step.

Definition 11. *Let $Q = \{q_1, ..., q_m\}$ be a set of modes. For any $q \in Q$, and $i \in \mathbb{N}$, use b_q^i to represent a Boolean variable. We now define*

$$\mathsf{enforce}_Q(q,i) = b_q^i \wedge \bigwedge_{p \in Q \setminus \{q\}} \neg b_p^i$$

$$\mathsf{enforce}_Q(q,q',i) = b_q^i \wedge \neg b_{q'}^{i+1} \wedge \bigwedge_{p\in Q\setminus\{q\}} \neg b_p^i \wedge \bigwedge_{p'\in Q\setminus\{q'\}} \neg b_{p'}^{i+1}$$

We omit the subscript Q when the context is clear.

The use of the auxiliary of formulas will be explained when we define the full encodings of bounded reachability.

Systems with no invariants. We start with the simplest case for hybrid automata with no invariants. Naturally, we say a hybrid automaton H is *invariant-free* if $\mathsf{inv}_q(H) = \top$ for every $q \in Q(H)$. We define the following formula that checks whether an unsafe region is reachable after exactly k steps of discrete transition in a hybrid system.

Definition 12 (k-Step Reachability, Invariant-Free Case). *Suppose H is invariant-free, and U a subset of its state space represented by* unsafe*. The $\mathcal{L}_{\mathbb{R}_\mathcal{F}}$-formula* $\mathsf{Reach}_{H,U}(k,M)$ *is defined as:*

$$\begin{aligned}
&\exists^X \boldsymbol{x}_0 \exists^X \boldsymbol{x}_0^t \cdots \exists^X \boldsymbol{x}_k \exists^X \boldsymbol{x}_k^t \exists^{[0,M]} t_0 \cdots \exists^{[0,M]} t_k.\\
&\bigvee_{q\in Q} \Big(\mathsf{init}_q(\boldsymbol{x}_0) \wedge \mathsf{flow}_q(\boldsymbol{x}_0, \boldsymbol{x}_0^t, t_0) \wedge \mathsf{enforce}(q,0)\Big)\\
\wedge \bigwedge_{i=0}^{k-1} \Bigg(&\bigvee_{q,q'\in Q} \Big(\mathsf{jump}_{q\to q'}(\boldsymbol{x}_i^t, \boldsymbol{x}_{i+1}) \wedge \mathsf{enforce}(q,q',i)\\
&\qquad\qquad \wedge \mathsf{flow}_{q'}(\boldsymbol{x}_{i+1}, \boldsymbol{x}_{i+1}^t, t_{i+1}) \wedge \mathsf{enforce}(q', i+1)\Big)\Bigg)\\
\wedge \bigvee_{q\in Q} &\mathsf{unsafe}_q(\boldsymbol{x}_k^t).
\end{aligned}$$

Intuitively, the trajectories start with some initial state satisfying $\mathsf{init}_q(\boldsymbol{x}_0)$ for some q. In each step, it follows $\mathsf{flow}_q(\boldsymbol{x}_i, \boldsymbol{x}_i^t, t)$ and makes a continuous flow from $\boldsymbol{x}_i$ to $\boldsymbol{x}_i^t$ after time t. When H makes a jump from mode q' to q, it resets variables following $\mathsf{jump}_{q'\to q}(\boldsymbol{x}_k^t, \boldsymbol{x}_{k+1})$. The auxiliary enforce formulas ensure that picking $\mathsf{jump}_{q\to q'}$ in the i-the step enforces picking flow'_q in the $(i+1)$-th step.

Systems with invariants and deterministic flows. When the invariants are not trivial, we need to ensure that for all the time points along a continuous flow, the invariant condition holds. Thus, we need to universally quantify over time. This is a fact that has been previously discussed, for instance, in [8]. However, if we allow nondeterministic flows, the situation is more complicated, which has not been discovered in existing work. We give the encoding for systems with only deterministic flows first, as follows:

Definition 13 (k-Step Reachability, Nontrivial Invariant and Deterministic Flow). *Suppose H contains invariants and only deterministic flow , and U a subset of its state space represented by*

unsafe. *In this case, the $\mathcal{L}_{\mathbb{R}_\mathcal{F}}$-formula* $\mathsf{Reach}_{H,U}(k, M)$ *is defined as:*

$$
\begin{aligned}
&\exists^X \boldsymbol{x}_0 \exists^X \boldsymbol{x}_0^t \cdots \exists^X \boldsymbol{x}_k \exists^X \boldsymbol{x}_k^t \exists^{[0,M]} t_0 \cdots \exists^{[0,M]} t_k.\\
&\bigvee_{q \in Q} \Big(\mathsf{init}_q(\boldsymbol{x}_0) \wedge \mathsf{flow}_q(\boldsymbol{x}_0, \boldsymbol{x}_0^t, t_0) \wedge \mathsf{enforce}(q, 0) \\
&\qquad\qquad \wedge \forall^{[0,t_0]} t \forall^X \boldsymbol{x}\ (\mathsf{flow}_q(\boldsymbol{x}_0, \boldsymbol{x}, t) \rightarrow \mathsf{inv}_q(\boldsymbol{x})) \Big) \\
\wedge &\bigwedge_{i=0}^{k-1} \Bigg(\bigvee_{q,q' \in Q} \Big(\mathsf{jump}_{q \rightarrow q'}(\boldsymbol{x}_i^t, \boldsymbol{x}_{i+1}) \wedge \mathsf{flow}_{q'}(\boldsymbol{x}_{i+1}, \boldsymbol{x}_{i+1}^t, t_{i+1}) \wedge \mathsf{enforce}(q, q', i) \\
&\qquad \wedge \mathsf{enforce}(q', i+1) \wedge \forall^{[0,t_{i+1}]} t \forall^X \boldsymbol{x}\ (\mathsf{flow}_{q'}(\boldsymbol{x}_{i+1}, \boldsymbol{x}, t) \rightarrow \mathsf{inv}_{q'}(\boldsymbol{x})) \Big) \Bigg) \\
\wedge &\bigvee_{q \in Q} (\mathsf{unsafe}_q(\boldsymbol{x}_k^t) \wedge \mathsf{enforce}(q, k)).
\end{aligned}
$$

The extra universal quantifier for each continuous flow expresses the requirement that for all the time points between the initial and ending time point ($t \in [0, t_i + 1]$) in a flow, the continuous variables $\boldsymbol{x}$ must take values that satisfy the invariant conditions $\mathsf{inv}_q(\boldsymbol{x})$.

Systems with invariants and nondeterministic flows. In the most general case, a hybrid system can contain non-deterministic flow: i.e., for some $q \in Q$, there exists $\boldsymbol{a}_0, \boldsymbol{a}_t, \boldsymbol{a}_t' \in \mathbb{R}^n$ and $t \in \mathbb{R}$ such that $\boldsymbol{a}_t \neq \boldsymbol{a}_t'$ and $\mathbb{R} \models \mathsf{flow}_q(\boldsymbol{a}_0, \boldsymbol{a}_t, t)$ and $\mathbb{R} \models \mathsf{flow}_q(\boldsymbol{a}_0, \boldsymbol{a}_t', t)$. Consequently, there is multiple possible values for the continuous variable for each time point. Different values correspond to different trajectories, and we only look for one of the trajectories that satisfies the invariant on all time points. Thus, we need to quantify over a trajectory and write $\exists \xi \forall t.\ \mathsf{inv}(\xi(t))$. We conjecture that, in general, this second-order quantification can not be fully reduced to a first-order expression.

In practice, the discussion of the invariant conditions in the existing work has (implicitly) assumed that the invariant condition should hold for all possible trajectories in the case of non-deterministic flow. We can formulate this assumption in the following way:

Definition 14 (Strictly-Imposed Invariants). *We say a hybrid automaton H has strictly-imposed mode invariants, if the following condition holds. Let* $\mathsf{flow}_q(\boldsymbol{x}, \boldsymbol{y}, t)$ *and* $\mathsf{inv}_q(\boldsymbol{x})$ *be the flow and invariant conditions in any mode q of H. Let $\boldsymbol{a}$ be an arbitrary starting point in the mode, satisfying* $\mathsf{inv}(\boldsymbol{a})$*. Then, for any $\boldsymbol{b}, \boldsymbol{b}' \in X(H)$ such that* $\mathsf{flow}(\boldsymbol{a}, \boldsymbol{b}, \tau)$ *and* $\mathsf{flow}(\boldsymbol{a}, \boldsymbol{b}', \tau)$ *are true at the same time point $\tau \in \mathbb{R}$, we have* $\mathsf{inv}_q(\boldsymbol{b})$ *iff* $\mathsf{inv}_q(\boldsymbol{b}')$.

If this condition is true, then a witness trajectory of bounded reachability has to require that all flows satisfy the same invariants. Consequently, we can still use the encoding in Definition 13, which requires that all possible flows satisfy the invariants. Thus, when this condition applies, we can still use first-order encoding for reachability in the presence of non-deterministic flows.

3.2 δ-Complete Analysis of Bounded Reachability

We now define the δ-complete analysis problem and prove its decidability.

Definition 15. *Let H be a hybrid system and U a subset of its state space. Suppose U is represented by the $\mathcal{L}_{\mathbb{R}_\mathcal{F}}$-formula* unsafe*. Let $k \in \mathbb{N}$ and $M \in \mathbb{R}^+$. The δ-complete analysis for (k, M)-bounded reachability problem asks for one of the following answers:*

- *(k, m)-Safety: H does not reach $[\![\mathsf{unsafe}]\!]$ within the (k, M)-bound.*
- *δ-Unsafety: H^δ reaches $[\![\mathsf{unsafe}^\delta]\!]$ within the (k, M)-bound.*

The following lemma comes from the intuitive meaning of the encodings. A proof is given in the appendix.

Lemma 1. *Let $\delta \in \mathbb{Q}^+ \cup \{0\}$ be arbitrary. Suppose H is a well-defined hybrid automaton with strictly-imposed invariants. Let U a subset of the state space of H, represented by the set* unsafe *of $\mathcal{L}_{\mathbb{R}_\mathcal{F}}$-formulas. Let $\mathsf{Reach}_{H,U}(k, M)$ be the $\mathcal{L}_{\mathbb{R}_\mathcal{F}}$-formula encoding (k, M)-bounded reachability of H with respect to U. We always have that $\mathbb{R} \models (\mathsf{Reach}_{H,U}(k, M))^\delta$ iff there exists a trajectory $\xi \in [\![H^\delta]\!]$ such that for some $(k, t) \in T(\xi)$, where $0 \le t \le M$, $(\xi(k, t), \sigma_\xi(k)) \in [\![\mathsf{unsafe}^\delta]\!]$.*

Now we can show that δ-complete analysis for bounded reachability problems is decidable for general $\mathcal{L}_{\mathbb{R}_\mathcal{F}}$-representable hybrid systems.

Theorem 5 (Decidability). *Let $\delta \in \mathbb{Q}^+$ be arbitrary. There exists an algorithm such that, for any bounded well-defined hybrid automaton $\mathcal{L}_{\mathbb{R}_\mathcal{F}}$-represented by H with strictly imposed invariants, and any unsafe region U $\mathcal{L}_{\mathbb{R}_\mathcal{F}}$-represented by* unsafe*, correctly performs δ-complete analysis for (k, M)-bounded reachability for H, for any $k \in \mathbb{N}, M \in \mathbb{R}^+$.*

Proof. We need to show that there is an algorithm that correctly returns one of the following:

- H does not reach $[\![\mathsf{unsafe}]\!]$ within the (k, M)-bound.
- H^δ reaches $[\![\mathsf{unsafe}^\delta]\!]$ within the (k, M)-bound.

To do this, we only need to solve the δ-decision problem of $\mathsf{Reach}_{H,U}(i, M)$ for $0 \le i \le k$. We obtain either $\mathsf{Reach}_{H,U}(i, M)$ is false for all such i, or is δ-true for some i, then:

- Suppose $\mathsf{Reach}_{H,U}(i, M)$ is false for all i. Then we know that for any $i \le k$, $\mathsf{Reach}_{H,U}(i, M)$ is false. Using Lemma 1 for the special case $\delta = 0$, we know that there does not exist a trajectory $\xi \in [\![H]\!]$ that can reach U within i steps, and consequently the system is safe within the (k, M)-bound.
- Suppose $\mathsf{Reach}_{H,U}(i, M)$ is δ-true for some i. We know that there exists $i \le k$ such that $\mathsf{Reach}^\delta_{H,U}(i, M)$ is true. Using Lemma 1 for $\delta \in \mathbb{Q}^+$, we know that there exists a trajectory $\xi \in [\![H^\delta]\!]$ that can reach the region represented by unsafe^δ in i-steps, i.e., within the (k, M)-bound. □

From the structures of the $\mathcal{L}_{\mathbb{R}_\mathcal{F}}$-formulas encoding δ-reachability, we can obtain the following complexity results of the reachability problems.

Theorem 6 (Complexity). *Suppose all the $\mathcal{L}_{\mathbb{R}_\mathcal{F}}$-terms in the description of H and U are in complexity class* C*. Then deciding the (k, M)-bounded δ-reachability problem is in*

- NP^C *for an invariant-free H;*

– $(\Sigma_2^P)^{\mathsf{C}}$ *for an* H *with strictly-imposed nontrivial invariants.*

Corollary 1. *For linear and polynomial hybrid automata,* δ*-complete bounded reachability analysis ranges from being* NP*-complete to* Σ_2^{P}*-complete for the three cases. For hybrid automata that can be* $\mathcal{L}_{\mathbb{R}_{\mathcal{F}}}$*-represented with whose* $\mathcal{F}$ *contains the set of ODEs defined* P*-computable right-hand side functions, the problem is* PSPACE*-complete.*

The results come from the fact that the complexity of polynomials is in P, and the set of ODEs in questions are PSPACE-complete.

Remark 4. The complexity results indicate that the worst-case running time of the analysis is exponential in all the input parameters. In particular, the worst-case running time grows exponentially with the δ and the size of the domains. We need to use efficient decision procedures to manage this complexity.

4 Experiments

Our tool dReach implements the techniques presented in the paper. The tool is built on several existing packages,including opensmt [6] for the general DPLL(T) framework, realpaver [16] for ICP, and CAPD [1] for computing interval-enclosures of ODEs. The tool is open-source at `http://dreal.cs.cmu.edu/dreach.html`. All benchmarks and data shown here are also available on the tool website.All experiments were conducted on a machine with a 3.4GHz octa-core Intel Core i7-2600 processor and 16GB RAM, running 64-bit Ubuntu 12.04LTS. Table 1 is a summary of the running time of the tool on various hybrid system models which we explain below.

Atrial Fibrillation. We studied the Atrial Fibrillation model as developed in [17]. The model has four discrete control locations, four state variables, and nonlinear ODEs. A typical set of ODEs in the model is:

$$\begin{aligned}
\frac{du}{dt} &= e + (u - \theta_v)(u_u - u)vg_{fi} + wsg_{si} - g_{so}(u) \\
\frac{ds}{dt} &= \frac{g_{s2}}{(1 + \exp(-2k(u - us)))} - g_{s2}s \\
\frac{dv}{dt} &= -g_v^+ \cdot v \qquad \frac{dw}{dt} = -g_w^+ \cdot w
\end{aligned}$$

The exponential term on the right-hand side of the ODE is the sigmoid function, which often appears in modelling biological switches.

Prostate Cancer Treatment. The Prostate Cancer Treatment model [23] exhibits more nonlinear ODEs. The reachability questions are

$$\begin{aligned}
\frac{dx}{dt} &= (\alpha_x(k_1 + (1-k_1)\frac{z}{z+k_2} - \beta_x((1-k_3)\frac{z}{z+k_4} + k_3)) - m_1(1 - \frac{z}{z_0}))x + c_1 x \\
\frac{dy}{dt} &= m_1(1 - \frac{z}{z_0})x + (\alpha_y(1 - d\frac{z}{z_0}) - \beta_y)y + c_2 y \\
\frac{dz}{dt} &= \frac{-z}{\tau} + c_3 z \\
\frac{dv}{dt} &= (\alpha_x(k_1 + (1-k_1)\frac{z}{z+k_2} - \beta_x(k_3 + (1-k_3)\frac{z}{z+k_4})) \\
&\quad -m_1(1 - \frac{z}{z_0}))x + c_1 x + m_1(1 - \frac{z}{z_0})x + (\alpha_y(1 - d\frac{z}{z_0}) - \beta_y)y + c_2 y
\end{aligned}$$

Electronic Oscillator. The EO model represents an electronic oscillator model that contains nonlinear ODEs such as the following:

$$\begin{aligned}
\frac{dx}{dt} &= -ax \cdot sin(\omega_1 \cdot \tau) \\
\frac{dy}{dt} &= -ay \cdot sin((\omega_1 + c_1) \cdot \tau) \cdot sin(\omega_2) \cdot 2 \\
\frac{dz}{dt} &= -az \cdot sin((\omega_2 + c_2) \cdot \tau) \cdot cos(\omega_1) \cdot 2 \\
\frac{\omega_1}{dt} &= -c_3 \cdot \omega_1 \quad \frac{\omega_2}{dt} = -c_4 \cdot \omega_2 \quad \frac{d\tau}{dt} = 1
\end{aligned}$$

Quadcopter Control. We developed a model that contains the full dynamics of a quadcopter. We use the model to solve control problems by answering reachability questions. A typical set of the

differential equations are the following:

$$\frac{\mathrm{d}\omega_x}{\mathrm{d}t} = L \cdot k \cdot (\omega_1^2 - \omega_3^2)(1/I_{xx}) - (I_{yy} - I_{zz})\omega_y\omega_z/I_{xx}$$
$$\frac{\mathrm{d}\omega_y}{\mathrm{d}t} = L \cdot k \cdot (\omega_2^2 - \omega_4^2)(1/I_{yy}) - (I_{zz} - I_{xx})\omega_x\omega_z/I_{yy}$$
$$\frac{\mathrm{d}\omega_z}{\mathrm{d}t} = b \cdot (\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2)(1/I_{zz}) - (I_{xx} - I_{yy})\omega_x\omega_y/I_{zz}$$
$$\frac{\mathrm{d}\phi}{\mathrm{d}t} = \omega_x + \frac{\sin(\phi)\sin(\theta)}{\left(\frac{\sin(\phi)^2\cos(\theta)}{\cos(\phi)} + \cos(\phi)\cos(\theta)\right)\cos(\phi)}\omega_y + \frac{\sin(\theta)}{\frac{\sin(\phi)^2\cos(\theta)}{\cos(\phi)} + \cos(\phi)\cos(\theta)}\omega_z$$
$$\frac{\mathrm{d}\theta}{\mathrm{d}t} = -\Big(\frac{\sin(\phi)^2\cos(\theta)}{\left(\frac{\sin(\phi)^2\cos(\theta)}{\cos(\phi)}\omega_y + \cos(\phi)\cos(\theta)\right)\cos(\phi)^2} + \frac{1}{\cos(\phi)}\Big)\omega_y - \frac{\sin(\phi)\cos(\theta)}{\left(\frac{\sin(\phi)^2\cos(\theta)}{\cos(\phi)} + \cos(\phi)\cos(\theta)\right)\cos(\phi)}\omega_z$$
$$\frac{\mathrm{d}\psi}{\mathrm{d}t} = \frac{\sin(\phi)}{\left(\frac{\sin(\phi)^2\cos(\theta)}{\cos(\phi)} + \cos(\phi)\cos(\theta)\right)\cos(\phi)}\omega_y + \frac{1}{\frac{\sin(\phi)^2\cos(\theta)}{\cos(\phi)} + \cos(\phi)\cos(\theta)}\omega_z$$
$$\frac{\mathrm{d}xp}{\mathrm{d}t} = (1/m)(\sin(\theta)\sin(\psi)k(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) - k \cdot d \cdot xp)$$
$$\frac{\mathrm{d}yp}{\mathrm{d}t} = (1/m)(-\cos(\psi)\sin(\theta)k(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) - k \cdot d \cdot yp)$$
$$\frac{\mathrm{d}zp}{\mathrm{d}t} = (1/m)(-g - \cos(\theta)k(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) - k \cdot d \cdot zp$$
$$\frac{\mathrm{d}x}{\mathrm{d}t} = xp, \frac{\mathrm{d}y}{\mathrm{d}t} = yp, \frac{\mathrm{d}z}{\mathrm{d}t} = zp$$

Room for Improvements. We aim to provide an open-source framework that allows much more optimization. In particular, while we can solve highly nonlinear models that are beyond the scope of other existing tools, there are simpler examples that other tools perform better. For instance, the Flow* tool [7] can efficiently compute a tight enclosure of the following system, while our tool does not terminate in reasonable time:

$$dx/dt = -9(x-2) - 7(y+2) + (z-1) + 0.2(x-2)(y+2) + 0.1(y+2)(z-1) + 0.1(x-2)(z-1) + 0.5(z-1)^2$$
$$dy/dt = 6(x-2) + 4(y+2) + z - 1$$
$$dz/dt = 3(x-2) + 2(y+2) - 2.5(z-1)$$

The reason is that the CAPD package that we use for verified integration of ODE blows up on this set of equations. However, our framework can integrate any reachable set computation tool, in replace of CAPD, for computing pruning on continuous flows. We remark on this in the next section.

Benchmark	#Mode	#Depth	#ODEs	#Vars	Delta	Result	Time(s)	Trace
AF-GOOD	4	3	20	53	0.001	SAT	0.425	793K
AF-BAD	4	3	20	53	0.001	UNSAT	0.074	—
AF-TO1-GOOD	4	3	24	62	0.001	SAT	2.750	224K
AF-TO1-BAD	4	3	24	62	0.001	UNSAT	5.189	—
AF-TO2-GOOD	4	3	24	62	0.005	SAT	3.876	553K
AF-TO2-BAD	4	3	24	62	0.001	UNSAT	8.857	—
AF-TSO1-TSO2	4	3	24	62	0.001	UNSAT	0.027	—
AF8-K7	8	7	40	101	0.001	SAT	10.478	3.8M
AF8-K23	8	23	40	293	0.001	SAT	135.29	11M
EO-K2	3	2	18	48	0.01	SAT	3.144	1.9M
EO-K11	3	11	99	174	0.01	UNSAT	0.969	—
QUAD-K1	2	1	34	89	0.01	SAT	2.386	10M
QUAD-K2	2	2	34	125	0.01	SAT	4.971	13M
QUAD-K3	4	3	68	161	0.01	SAT	13.755	42M
QUAD-K3U	4	3	68	161	0.01	UNSAT	2.846	—
CT	2	2	10	41	0.005	SAT	345.84	3.1M
CT	2	2	10	41	0.002	SAT	362.84	3.1M
BB-K10	2	10	22	66	0.01	SAT	8.057	123K
BB-K20	2	20	42	126	0.01	SAT	39.196	171K

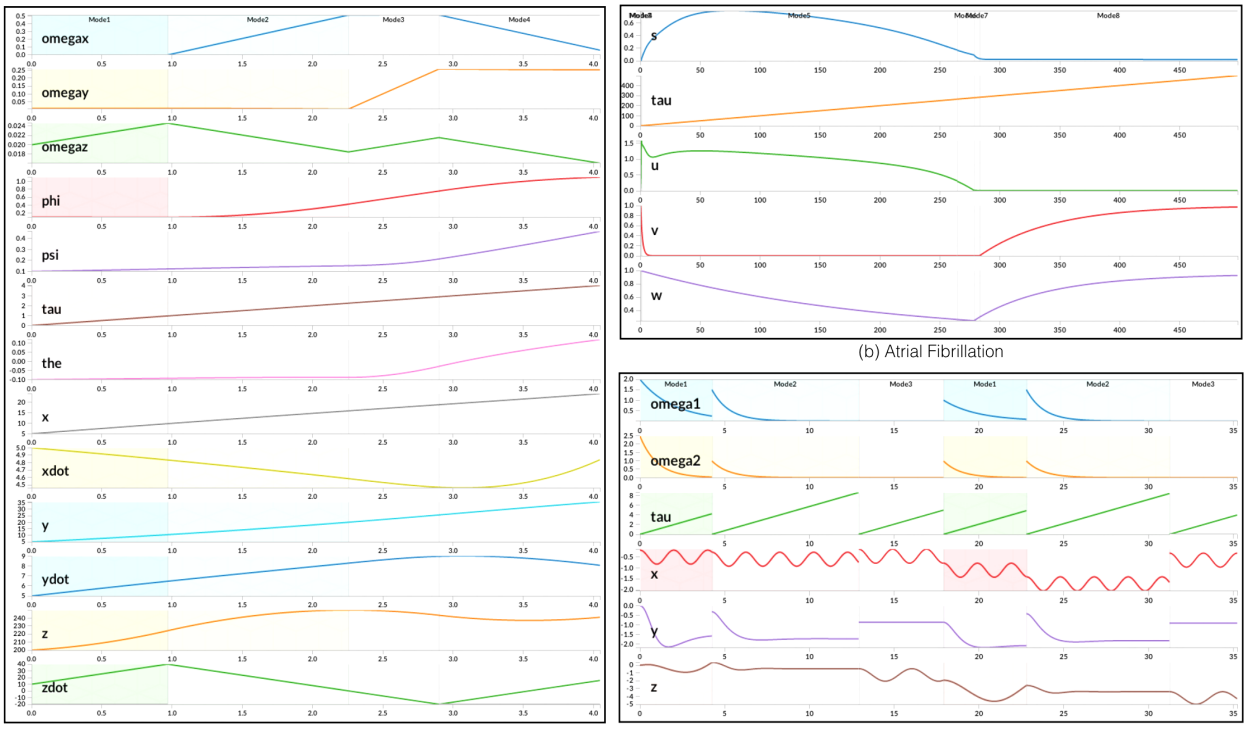
Table 1: #Mode = Number of modes in the hybrid system, #Depth = Unrolling depth, #ODEs = Number of ODEs in the unrolled formula, #Vars = Number of variables in the unrolled formula, Result = Bounded Model Checking Result (delta-SAT/UNSAT) Time = CPU time (s), Trace = Size of the ODE trajectory, AF = Atrial Filbrillation, EO = Electronic Oscillator, QUAD = Quadcopter Control, CT = Cancer Treatment, BB = Bouncing Ball with Drag.

5 Discussion

Reachable set computation, which computes geometric representations of the complete set of reachable states, is the mainstream approach for analyzing bounded reachability of hybrid systems. The techniques can have difficulty in scaling on systems with very complex dynamics and discrete transitions. Bounded model checking has the advantage of focusing the search for one counterexample, and does not maintain the complete set of reachable states. With fast SAT/SMT solvers, bounded model checking techniques can natively handle the discrete components in hybrid systems. Bounded model checking requires a very powerful solver, one that can handle ODEs and nested quantifiers. We have proved that the complexity of bounded δ-reachability is comparable to SAT solving, and it is reasonable to expect that with more improvement on the solver, large realistic systems can eventually be handled in practice. Note again that all the techniques in reachable set computation can be directly used in logic solvers, and it is possible to have practical tools that combine the advantages of both approaches.

6 Conclusion

We developed the framework of δ-complete analysis for bounded reachability of a wide range of hybrid systems. δ-Complete reachability analysis reduces verification problems to δ-decision problems of formulas over the reals. It follows from δ-decidability of these formulas that δ-complete



(a) Quadcopter Control

(b) Atrial Fibrillation

(c) Electronic Oscillator

Fig. 1: Example trajectories computed for the following models: (a) Quadcopter Control, (b) Atrial Fibrillation, (c) Electronic Oscillator.

reachability analysis of a wide range of nonlinear hybrid systems is decidable. In practice, δ-reachability problems are solved through reduction to δ-decision problems for first-order formulas over the reals. We demonstrated the scalability of our approach on highly nonlinear hybrid systems.

References

1. CAPD: Computer assisted proofs in dynamical systems. `http://capd.ii.uj.edu.pl/index.php`.
2. R. Alur. Formal verification of hybrid systems. In *EMSOFT*, pages 273–278, 2011.
3. R. Alur, C. Courcoubetis, T. A. Henzinger, and P.-H. Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, editors, *Hybrid Systems*, volume 736 of *Lecture Notes in Computer Science*, pages 209–229. Springer, 1992.
4. R. Alur and D. L. Dill. The theory of timed automata. In J. W. de Bakker, C. Huizing, W. P. de Roever, and G. Rozenberg, editors, *REX Workshop*, volume 600 of *Lecture Notes in Computer Science*, pages 45–73. Springer, 1991.
5. G. Audemard, M. Bozzano, A. Cimatti, and R. Sebastiani. Verifying industrial hybrid systems with mathsat. *Electr. Notes Theor. Comput. Sci.*, 119(2):17–32, 2005.
6. R. Bruttomesso, E. Pek, N. Sharygina, and A. Tsitovich. The opensmt solver. In J. Esparza and R. Majumdar, editors, *TACAS*, volume 6015 of *Lecture Notes in Computer Science*, pages 150–153. Springer, 2010.
7. X. Chen, E. Ábrahám, and S. Sankaranarayanan. Taylor model flowpipe construction for non-linear hybrid systems. In *RTSS*, pages 183–192, 2012.
8. A. Cimatti, S. Mover, and S. Tonetta. A quantifier-free SMT encoding of non-linear hybrid automata. In *FMCAD*, pages 187–195, 2012.

9. J. M. Davoren. Epsilon-tubes and generalized skorokhod metrics for hybrid paths spaces. In *HSCC*, pages 135–149, 2009.
10. M. Fränzle. Analysis of hybrid systems: An ounce of realism can save an infinity of states. In J. Flum and M. Rodríguez-Artalejo, editors, *CSL*, volume 1683 of *Lecture Notes in Computer Science*, pages 126–140. Springer, 1999.
11. M. Fränzle, T. Teige, and A. Eggers. Engineering constraint solvers for automatic analysis of probabilistic hybrid automata. *J. Log. Algebr. Program.*, 79(7):436–466, 2010.
12. S. Gao, J. Avigad, and E. M. Clarke. Delta-complete decision procedures for satisfiability over the reals. In B. Gramlich, D. Miller, and U. Sattler, editors, *IJCAR*, volume 7364 of *Lecture Notes in Computer Science*, pages 286–300. Springer, 2012.
13. S. Gao, J. Avigad, and E. M. Clarke. Delta-decidability over the reals. In *LICS*, pages 305–314, 2012.
14. S. Gao, S. Kong, and E. M. Clarke. dreal: An smt solver for nonlinear theories over the reals. In *CADE*, pages 208–214, 2013.
15. S. Gao, S. Kong, and E. M. Clarke. Satisfiability modulo odes. In *FMCAD*, pages 105–112, 2013.
16. L. Granvilliers and F. Benhamou. Algorithm 852: Realpaver: an interval solver using constraint satisfaction techniques. *ACM Trans. Math. Softw.*, 32(1):138–156, 2006.
17. R. Grosu, G. Batt, F. H. Fenton, J. Glimm, C. L. Guernic, S. A. Smolka, and E. Bartocci. From cardiac cells to genetic regulatory networks. In *CAV*, pages 396–411, 2011.
18. S. Gulwani and A. Tiwari. Constraint-based approach for analysis of hybrid systems. In A. Gupta and S. Malik, editors, *CAV*, volume 5123 of *Lecture Notes in Computer Science*, pages 190–203. Springer, 2008.
19. T. A. Henzinger. The theory of hybrid automata. In *LICS*, pages 278–292, 1996.
20. T. A. Henzinger and J.-F. Raskin. Robust undecidability of timed and hybrid systems. In *HSCC*, pages 145–159, 2000.
21. C. Herde, A. Eggers, M. Fränzle, and T. Teige. Analysis of hybrid systems using hysat. In *ICONS*, pages 196–201, 2008.
22. Z. Huang and S. Mitra. Computing bounded reach sets from sampled simulation traces. In *HSCC*, pages 291–294, 2012.
23. B. Liu, S. Kong, S. Gao, and E. Clarke. Parameter identification using delta-decisions for biological hybrid systems. CMU SCS Technical Report, CMU-CS-13-136, 2014.
24. P. Prabhakar, V. Vladimerou, M. Viswanathan, and G. E. Dullerud. Verifying tolerant systems using polynomial approximations. In *RTSS*, pages 181–190, 2009.
25. S. Ratschan. Safety verification of non-linear hybrid systems is quasi-semidecidable. In *TAMC*, pages 397–408, 2010.
26. K. Weihrauch. *Computable Analysis: An Introduction*. 2000.

Appendix

Proof of Lemma 1.

Proof. We prove for the case with nontrivial invariants. We work with the unperturbed encoding, which easily applies to the δ-perturbed version. We will need to do induction on the subformula of $\mathsf{Reach}_{H,U}$ that does not contain the unsafe conditions. For reasons that will be made clear below, we split the formula $\mathsf{Reach}_{H,U(k,M)}$ into two parts and write it as the conjunction $\mathsf{traj}(k,M) \wedge \mathsf{unsafe}(k)$, where $\mathsf{unsafe}(k)$ is $\bigvee_{q\in Q}(\mathsf{unsafe}_q(\boldsymbol{x}_k^t) \wedge \mathsf{enforce}(q,k))$.

Suppose $\mathbb{R} \models \mathsf{Reach}_{H,U}(k,M)$. We do induction on k to prove that there exists a trajectory $\xi \in [\![H]\!]$ that contains k mode changes. When $k = 0$, without loss of generality we pick an arbitrary starting mode q, such that the $\mathsf{traj}(k,M)$ part of the formula can be simplified as

$$\exists^X \boldsymbol{x}_0 \exists^X \boldsymbol{x}_0^t \exists^{[0,M]} t_0 \Big(\mathsf{init}_q(\boldsymbol{x}_0) \wedge \mathsf{flow}_q(\boldsymbol{x}_0, \boldsymbol{x}_0^t, t_0) \wedge \mathsf{enforce}(q,0) \\ \wedge \forall^{[0,t_0]} t \forall^X \boldsymbol{x}\ (\mathsf{flow}_q(\boldsymbol{x}_0, \boldsymbol{x}, t) \rightarrow \mathsf{inv}_q(\boldsymbol{x}))\Big).$$

Since the formula is true, there exists witnesses $\boldsymbol{a}, \boldsymbol{a}^t, \tau$ such that the quantifier-free part is satisfied. By well-definedness of flow_q there exists a trajectory ξ from $\boldsymbol{a}_0$ to $\boldsymbol{a}^t$ such that for any $0 \le \tau' \le \tau$, $\xi(\tau)$ satisfies the invariant condition. Now, suppose $k = (k-1)+1$ ($k \ge 1$) and by inductive hypothesis there exists a trajectory $\xi' \in [\![H]\!]$ with $k-1$ mode changes. We now extend ξ' with one more mode change. Let $\mathsf{traj}(k-1,M)$ be the part of $\mathsf{Reach}_{H,U})(k-1,M)$, and thus $\mathsf{traj}(k,M)$ can be written as

$$\exists \boldsymbol{x}_k \exists^X \boldsymbol{x}_k^t \exists^{[0,M]} t_k \\ \Bigg(\mathsf{traj}(k-1,M) \wedge \bigvee_{q,q'\in Q} \Big(\mathsf{jump}_{q\rightarrow q'}(\boldsymbol{x}_{k-1}^t, \boldsymbol{x}_k) \wedge \mathsf{flow}_{q'}(\boldsymbol{x}_k, \boldsymbol{x}_k^t, t_k) \\ \wedge \mathsf{enforce}(q,q',i) \wedge \forall^{[0,t_k]} t \forall^X \boldsymbol{x}\ (\mathsf{flow}_{q'}(\boldsymbol{x}_k, \boldsymbol{x}, t) \rightarrow \mathsf{inv}_{q'}(\boldsymbol{x}))) \wedge \mathsf{enforce}(q',k)\Big)\Bigg)$$

Note that $\boldsymbol{x}_0, ..., \boldsymbol{x}_{k-1}^t$ are quantified variables in $\mathsf{traj}(k-1,M)$. Since the formula is true, there exists $\boldsymbol{a}_k, \boldsymbol{a}_k^t, \tau_k$ that witness the satisfiability of the quantifier-free part of the formula outside of $\mathsf{traj}(k-1,M)$. Now, we extend $\xi' \in [\![H]\!]$ in the following way. Let the last state of ξ' be given by $\boldsymbol{a}_{k-1}^t$. Following the formula, we have that $\mathsf{jump}_{1\rightarrow q'}(\boldsymbol{a}_{k-1}^t, \boldsymbol{a}_k)$ satisfies the jumping condition between mode q and q'. It is then followed by a continuous trajectory that starts from $\boldsymbol{a}_k$ and ends at $\boldsymbol{a}_k^t$, satisfying $\mathsf{flow}(\boldsymbol{a}_k, \boldsymbol{a}_k^t, \tau_k)$. Thus, there exists a trajectory $\xi \in [\![H]\!]$ with k mode changes. Thus, for all k there exists a trajectory $\xi \in [\![H]\!]$ such that for some $(k,t) \in T(\xi)$, $\xi(k,t), \sigma_\xi(k) \in [\![\mathsf{unsafe}]\!]$.

The reverse direction is easy. Suppose there exists a trajectory $\xi \in [\![H]\!]$ such that for some $(k,t) \in T(\xi)$, $\xi(k,t), \sigma_\xi(k) \in [\![\mathsf{unsafe}]\!]$, then the start and end points in each piece of the continuous trajectories witness the formula $\mathsf{Reach}_{H,U}(k,M)$. □